# *Simple Orchestration Application Framework to Control "Burning Plasma Integrated Code"*

Takayuki Tatekawa, Kohei Nakajima, Naoya Teshima,
Guehee Kim, Yoshio Suzuki, Hiroshi Takemiya
Center for Computational Science and e-Systems,
Japan Atomic Energy Agency
6-9-3 Higashiueno, Taito-ku, Tokyo 110-0015, Japan
{tatekawa.takayuki, nakajima.kohei, teshima.naoya,
kim.guehee, suzuki.yoshio, takemiya.hiroshi}@jaea.go.jp

Nobuhiko Hayashi
Naka Fusion Institute,
Japan Atomic Energy Agency
801-1 Mukoyama, Naka-shi, Ibaraki 311-0193, Japan

Katsuyuki Iba
Research Organization for Information Science and Technology,
2-4, Shirane, Shirakata, Tokai-mura, Naka-gun, Ibaraki,
319-1106, Japan

*Abstract*— **We have developed the Simple Orchestration Application Framework (SOAF) on a grid infrastructure to control cooperative and multiple execution of simulation codes on remote computers from a client PC. SOAF enables researchers to generate a scenario of their cooperative and multiple executions by only describing a configuration file which includes the information of execution codes and file flows among them. SOAF does not need substantial modification of the simulation codes.**

**We have applied SOAF to the "Burning Plasma Integrated Code" which consists of various plasma simulation codes. In order to predict and interpret the behavior of fusion burning plasma, it is necessary to cooperatively and concurrently execute various simulation codes to understand complex plasma phenomena with wide temporal and spatial ranges. Those codes exist on distributed heterogeneous computers located in different sites such as universities and institutes. By using SOAF, we succeeded to cooperatively and concurrently execute four plasma simulation codes without substantial modification as described in the configuration file.**

***Keywords- grid computing, numerical simulation, orchestration, fusion, AEGIS***

## I. INTRODUCTION

Integrating various simulation codes, we have been able to carry out large-scale and detailed simulations. For example, in nuclear field, in order to predict and interpret the behavior of fusion burning plasma, it is necessary to simulate complex plasma phenomena with wide temporal and spatial ranges. There has been a problem how to execute the simulation codes on distributed computers cooperatively and concurrently.

A grid computing technology has been used to realize their cooperative and multiple executions. For example, a workflow tool such as Kepler [1], TME [2], and so on enables researchers to build and execute a scientific scenario from a client PC. Researchers can promote an execution of simulation codes and can visualize analysis processes using a graphical user interface (GUI). Here, researchers can construct the scenario by simple procedures such as drag and drop. A remote procedure call (RPC) enables researchers to cooperatively and concurrently manage various simulation codes by developing an application with its functions. GridRPC [3] is the expansion of RPC for a grid computing environment. An extended message passing interface (MPI) suitable to a grid environment (Grid-enabled MPI) enables a communication between simulation codes on heterogeneous distributed computers. Various types of grid-enabled MPI, STAMPI [4], MPICH-G [5], PACX MPI [6], and so on, have been developed. Researchers can cooperatively and concurrently execute simulation codes by modifying those codes.

Although researchers can cooperatively and concurrently execute various simulation codes by using those technologies, those technologies have advantages and disadvantages. We discuss them by roughly classifying types of the cooperative and multiple execution of simulation codes into three; type 1 is the workflow type in which the codes are executed sequentially, type 2 is the pipeline type in which simulation codes are executed in parallel by sending and receiving input/output data during their running, type 3 is the conditional branch type in which simulation codes are started and data are transferred depending on various conditions. A workflow tool is useful for type 1, but is not forte for type 2 and 3, since it does not always have a function for a detailed control such as conditional branch. On the other hand, GridRPC and Grid-enabled MPI are suitable for type 2 and 3. However, researchers have to make exertions to develop a grid-enabled application and/or to modify their simulation codes.

The integration of plasma simulation codes, e.g. the "Burning Plasma Integrated Code", corresponds to type 3, since those codes have to be cooperatively and concurrently executed depending on various conditions such as plasma stability, timing of heating, and so on. Especially the integrated code is "dynamical conditional branch" type. A workflow tool cannot operate this type. Therefore, it is desirable to construct a novel grid computing technology which enables researchers in plasma physics field to control the type 3 execution without their exertions.

As a novel technology to easily control various types of cooperative and multiple executions, we propose the Simple Orchestration Application Framework (SOAF). Using SOAF, researchers can control various types of cooperative and multiple executions of simulation codes by just describing the data dependency among those codes.

We have developed SOAF by using the client application program interface (API) for grid applications which has been implemented on Atomic Energy Grid Infrastructure (AEGIS) [7]. And we have applied SOAF to the "Burning Plasma Integrated Code" to solve the current diffusion, stability of plasma, current drive, and so on.

We describe SOAF in detail and its application to the "Burning Plasma Integrated Code" in section II and III, respectively. In section IV, we report the environment and result of our experiment. Finally in section V, we summarize our R&D results and describe a future work.

## II. Simple Orchestration Application Framework (SOAF)

### A. Overview of SOAF

We propose a novel framework to cooperatively and concurrently execute simulation codes without difficulty. Here, we describe the concept of framework.

We suppose that many simulation codes exist on distributed computers. Each code can analyze one phenomenon in detail. Integrating these codes, a realistic problem can be elucidated. To realize this, the orchestration including code executions and file transfers on a grid infrastructure is required. Here, we consider followings are the critical issue to design the SOAF: (1) How are various types of cooperative and multiple executions controlled easily? (2) How is each simulation executed cooperatively and concurrently with its minimal modification? (3) How are researchers' exertions for cooperative and multiple executions reduced? (4) Furthermore, how is the overhead due to the framework minimized?

We focused on a file flow among execution codes to design the SOAF. Firstly, it is easier to define a file flow than a flow of execution codes in order to build a scientific scenario depending on conditional branch. Secondly, it is better to send/receive information by transferring files in order to have communications among simulation codes without modifying those codes. We adopted the way to manage the start of codes by a file flow. It is useful to identify a file flow, since a code is usually started after it receives a file from another code, except a firstly started code. SOAF manages the start of codes and the transfer of files under a file flow which is described in a configuration file. SOAF consists of a client application, programs to support the file transfers, and a configuration file. Those programs (we call this program “sentinel”) are started by the client application (we call this "controller") and are executed on distributed computers. By those ideas, researchers can integrate various kinds of simulation codes by just describing the data dependency among these codes. We consider that a new code is executed with the trigger of output files from other codes. Only a little modification about file output is needed. The last issue is verified by an experiment.

### B. Controller

We developed the controller using the client API for grid application implemented on AEGIS. AEGIS is grid middleware for atomic energy research which we developed. The schematic diagram of AEGIS is shown in Figure 1. We developed the client API as a function of AEGIS to develop grid-enabled applications on a client PC. The client API can work on other grid middleware such as UNICORE, DIET, Globus and so on [8]. The client API is classified into low, middle and high level APIs due to their functions. To develop controller, we used the authentication API, file transfer API, job submission API and job information request API in low level APIs, and Job-script generator API in middle level APIs.

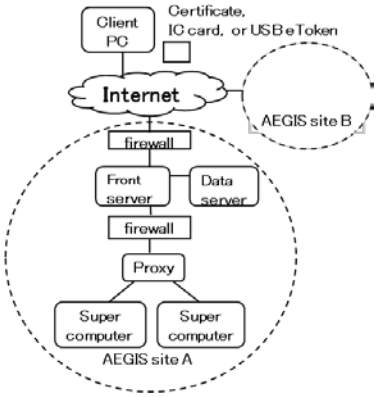


Figure 1. Schematic diagram of AEGIS.

Low level APIs supply connection between a client PC and supercomputers on AEGIS, job operation to supercomputers, resource handling on both clients PC and supercomputers, and so on. When users access AEGIS, users need an authentication process with an IC card or an USB token (PKCS#11). Job-script generator API generates the script corresponding to heterogeneous computers by reading job attributes such as name of computer, job class, number of CPUs, path of program, work directory, and so on.

### C. Sentinel

It is needed for the controller to confirm the end of file output. Thus, we developed the sentinel which detects the output files and operates the file transfer between simulation codes. It works before and after job submission.

The sentinel has three types of scripts; `send`, `recv`, and `void`. The `send` script detects "flag files" of output files, generates "sent files". The `recv` script detects "sent files" and generates "flag files". The `void` script is used to execute simulation codes without files.

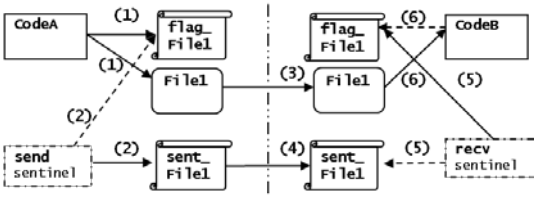


Figure 2. The procedure of file transfer

We show the procedure of file transfer using the sentinel. Here, we describe an example that two codes ("code A" and "code B") are cooperatively and concurrently executed on distributed computers (see Figure 2). The execution of "code B" requires output files from "code A". The procedure is as follows:

(1) "Code A" generates output files and their "flag files".

(2) The `send` sentinel beside "code A" detects the "flag files" and deletes them. Then `send` sentinel generates "sent files" of the output files.

(3) The controller transfers the output files from work directory of "code A" to that of "code B".

(4) The controller transfers "sent files" from work directory of "code A" to that of "code B".

(5) The `recv` sentinel beside "code B" detects "sent files" and deletes them. Then the `recv` sentinel generates "flag files" of transferred files.

(6) "Code B" is started by SOAF, detects "flag files" and read the output files.

A little modification of simulation codes is needed to use SOAF. When "code A" generates output files, "code A" must generate their "flag files". "Code B" must detect the "flag files".

*D. Configuration file*

The configuration file in this case is described in Figure 3. The configuration file consists of information of codes (PROGRAM) and file flow (FLOW). The "type" in FLOW represents kinds of code. The "type 1" corresponds to the firstly started code. The code with "type 0" is started after it receives a file from another code. The "File1" in FLOW is output files from code A. By the SOAF, these files are transferred from work directory of code A to that of code B. After files are transferred, the controller starts code B. The File2 is output file from code B. After execution of code B, this file is transferred from work directory of code B to that of code A. Code A receives this file.

```
PROGRAMNUM 2
PROGRAM
   program  CodeA
   name     CodeA_0
   server   computer1.node.site
   path     /home/foo/bin/CodeA
   work     /home/foo/workdir
   queue    TSS
   para     1
   kind     serial
END

PROGRAM
   program  CodeB
   name     CodeB_0
   server   computer2.node.site
   path     /home/foo/bin/CodeB
   work     /home/foo/workdir
   queue    TSS
   para     1
   kind     serial
END


FLOWNUM 2
FLOW
   alias    CodeA_0
   exec     1
   type     1
   send     File1    CodeB_0
   recv     File2    CodeB_0
END

FLOW
   alias    CodeB_0
   exec     1
   type     0
   recv     File1    CodeA_0
   send     File2    CodeA_0
END
```

Figure 3. An example of the configuration file. The overhead of the sentinel is almost negligible

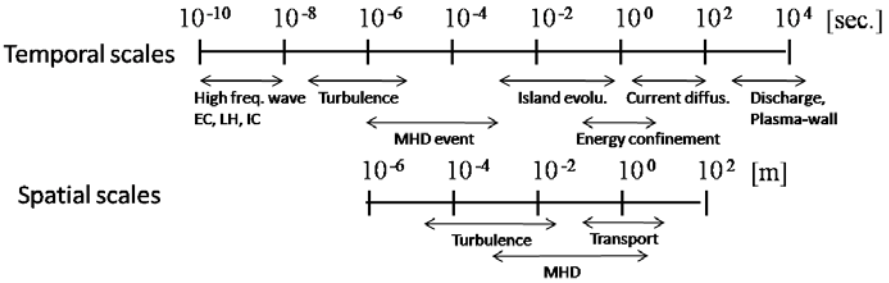


Figure 4. Temporal and spatial scales on physical processes of plasma

## III. BURNING PLASMA INTEGRATED CODE

The "Burning Plasma Integrated Code" has been researched and developed mainly by Naka Fusion Institute of Japan Atomic Energy Agency (JAEA) to integratedly predict and interpret the plasma behavior. It consists of various plasma simulation codes to solve the current diffusion, stability of plasma, current drive, and so on. A quite realistic simulation is expected by integrating those codes.

It is indispensable to understand the controllability of plasma toward the continuous operation of tokamak reactor especially for ITER [9]. To control the burning plasma and achieve high performance, it has an important role to simulate behavior of burning plasma in tokamak reactor.

For simulation of burning plasma in tokamak reactor, it is not realistic to handle whole physical processes by one simulation code. One of the reasons is that burning plasma has very wide temporal and spatial ranges in the steady state (Figure 4).

The burning plasma has complex physics. Each physical process with different temporal and spatial scales is modeled and calculated by separated simulation codes. The respective simulation codes can be integrated for the burning plasma analysis. The integrated simulation system covers both

microscopic and macroscopic physical processes and simulates long-time behavior considering short-time behavior.

For integrated simulation, we need a grid computing technology to combine loosely coupling existing codes on the Grid. Therefore, we have applied SOAF to manage those codes in the "Burning Plasma Integrated Code" on our grid infrastructure AEGIS. In the current application, we use four plasma simulation codes; tokamak prediction and interpretation code system (TOPICS) [10], two-dimensional magnetic stability analysis code (MARG2D) [11], electron-cyclotron current drive code (ECCD) [12], and lower-hybrid current drive code (LHCD).

TOPICS solves the 1D transport and current diffusion equations and the Grad-Shafranov equation of the MHD equilibrium on the 2D plane. The transport code solves the current diffusion equation, including EC and LH current profiles. TOPICS investigates specific characteristics of burning plasma such as behavior of edge localized modes (ELMs) [13] and neoclassical tearing modes (NTM) stability.

MARG2D is 2-D Newcomb equation solver which solves an eigenvalue problem associated with the two-dimensional Newcomb equation in axisymmetric toroidal plasma such as tokamak by using a finite element method. Using this code, we can analyze stability of ideal MHD modes from low to high toroidal mode numbers. Furthermore we obtain eigenfunctions numerically which show the singular behavior. Using MARG2D, the MHD property of JT-60SA, the complemented device of ITER, is investigated [14].

ECCD and LHCD codes simulate control and stabilization of the burning plasma and thus are executed for control of burning plasma simulated by TOPICS.

MARG2D, ECCD, and LHCD are started depending on the requirement arising from TOPICS during the burning plasma simulation. When the plasma is found to be close to the unstable region by MARG2D, TOPICS requests to start ECCD or LHCD for stabilization of the burning plasma. Therefore the "Burning Plasma Integrated Code" belongs to the conditional branch type. Total length of the simulation codes is about 300,000 lines.

## IV. EXPERIMENT

In this section, we mention our experiment about the application of SOAF to those four codes in "Burning Plasma Integrated Code".

The client PC where the controller is executed is Ubuntu Linux 8.04 located in Center for Computational Science and e-Systems of Japan Atomic Energy Agency (CCSE/JAEA) (Tokyo/Japan). The C compiler is gcc-4.2.4 (multithread enabled). We used USB token (PKCS#11) for authentication to AEGIS. The numerical simulations are submitted to three computers located in Tokai Research and Development Center of JAEA (Ibaraki/Japan); Altix3700Bx2, Altix350, and PC cluster. In this experiment, we fix the computers in which simulation codes are executed as shown in Table I. In this experiment, TOPICS, LHCD, and ECCD are executed on TSS mode (serial execution). MARG2D is submitted to job queuing system (class of 32CPUs and 3hours). When we start the controller, not only TOPICS but also each sentinel script beside simulation code is executed. All sentinel scripts are executed on TSS mode.

TABLE I. CODES AND COMPUTERS.

| Codes | Computers |
|---|---|
| TOPICS | Altix350 |
| MARG2D | Altix3700Bx2 |
| LHCD | Altix350 |
| ECCD | PC Cluster |

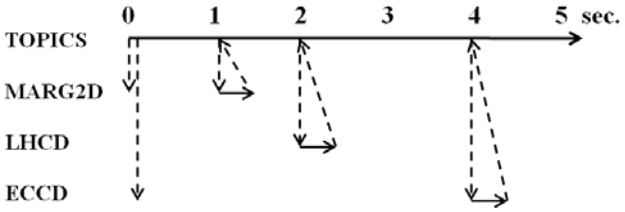


Figure 5. The diagram of the simulations. The solid line and dashed line mean code execution and file transfer, respectively.

The diagram of the simulations is shown in Figure 5. Those simulation codes are executed by the following procedure. At first, TOPICS is started. In this experiment, the simulation time (not CPU time) is set to 5 seconds (t=5). During the running of TOPICS, TOPICS requests to start MARG2D, LHCD, and ECCD. In the configuration file of controller, TOPICS corresponds to "type 1" code. MARG2D, LHCD, and ECCD correspond to "type 0" codes. During simulation by MARG2D, LHCD, and ECCD, TOPICS suspends. Then TOPICS reflects the analyses results by other codes and restarts.

The file flow is shown in Figure 6. The files are transferred between TOPICS and other 3 codes. MARG2D, LHCD, and ECCD receive the input files from TOPICS and return results of analyses.

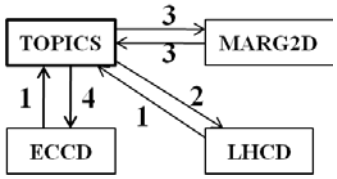


Figure 6. The file flow between simulation codes. The number beside arrow means the number of file.

We prepared the configuration file for four simulation codes (about 120 lines) to execute the controller. We confirmed that the controller executes those simulation codes as scheduled. SOAF successfully controls the cooperative and multiple executions of four simulation codes and file transfers between those codes.

The performance of our experiment is shown in Table II. The whole execution time is about 40 minutes without job queuing time of MARG2D. The overhead of controller is about 2 minutes and does not almost influence to the total execution time. Because the most of overhead is the authentication to AEGIS, the overhead is not considered to

increase substantially even if a cooperative and multiple executions of codes become complicated.

The timing of start for ECCD and LHCD is scheduled by the conditional branch implemented in TOPICS. Because the start of ECCD and LHCD is decided by only the existence of output files from TOPICS, SOAF is applicable for any conditional branch.

By using SOAF, we can execute cooperatively and concurrently various simulation codes on remote computers from a client PC without difficulty. The overhead for controller is much less than execution time of simulations. We can achieve the orchestration of simulations with minimal modification. For the he "Burning Plasma Integrated Code", the length of the modification is about 200 lines. We have verified that all of our issues are solved.

TABLE II. THE PERFORMANCE OF OUR EXPERIMENT. WALL-CLOCK TIME DOES NOT INCLUDE JOB QUEUING TIME.

| Wall-clock time [min.] | Action |
|---|---|
| 0 | SOAF start<br>TOPICS start |
| 4 | MARG2D start |
| 5 | MARG2D finish |
| 25 | LHCD start |
| 26 | LHCD finish |
| 33 | ECCD start |
| 36 | LHCD finish |
| 40 | TOPICS finish<br>SOAF finish |

## V. SUMMARY

We have developed a framework of SOAF which supports the development of large-scale and detailed simulation codes by loosely coupling existing codes on the Grid. In order to notify the timing of file transfer to SOAF, only a few SOAF library calls are inserted in the applications. We have confirmed the usefulness of SOAF by applying it to the "Burning Plasma Integrated Code". In the current experiment, we use four simulation codes on distributed computers. SOAF can control cooperative and multiple executions of these four simulation codes and file transfers between them. We have solved four issues described in Section II. Although we show an example for the conditional branch type, researcher can integrate various kinds of simulation codes by just describing the data dependency among these codes. Therefore, the first issue is solved. In order to notify the timing of file transfer to SOAF, only a few SOAF library calls are inserted in the applications. The second and third issues are solved. In our experiment, the overhead for SOAF is much less than that for simulations. Thus, the fourth issue is solved.

For consideration of the realistic situation, we need longer-time simulation. In the current experiment, the simulation time of burning plasma is only 5 seconds. For example, the burn duration in ITER project is designed as more than 1000 seconds [9]. When burning plasma in such a situation is simulated, we would require several weeks or several months. To achieve this, we need to consider the continuity of execution. The longer time simulation may suffers from various unexpected stop which is caused by execution time excess, queuing timeout, outage of computers and so on. To avoid the stop of the simulation, we now try to implement the fault-tolerant mechanism to SOAF. For example, even if the simulation code exhausts execution time, SOAF detects error of execution time excess and resubmits the code to restart. When this mechanism is implemented, a long-time simulation can be executed automatically.

## ACKNOWLEDGMENT

We would like to thank Dr. T. Ozeki for fruitful discussion and support and Mr. I. Kamata for modification of the simulation codes.

## REFERENCES

[1] B. Ludäscher *et al.*, “Scientific workflow management and the Kepler system”, Concurrency and Computation: Practice and Experience, vol.18, pp. 1039-1065, 2005.

[2] T. Imamura, Y. Hasegawa, N. Yamagishi, and H. Takemiya, “TME: A Distributed resource handling tool.", Recent Advances in Computational Science & Engineering, International Conference on Scientific & Engineering Computation (IC-SEC) (3-5 December 2002, Raffles City Convention Centre, Singapore), pp. 789-792, 2002.

[3] K. Seymour *et al.*, “Overview of GridRPC: A Remote Procedure Call API for Grid Computing”, Proceedings of 3rd International Workshop on Grid Computing (M. Parashar eds.), pp 274-278, 2002.

[4] T. Imamura, Y. Tsujita, H. Koide, and H. Takemiya, “An Architecture of Stampi: MPI Library on a Cluster of Parallel Computers”, Proceedings of the 7th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, Lecture Notes in Computer Science, vol. 1908, pp. 200-207, January 2000.

[5] I. Foster and N. T. Karonis, “A grid-enabled MPI: message passing in heterogeneous distributed computing systems”, Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM), pp.1-11, 1998.

[6] T. Beisel, E. Gabriel, and M. Resch, "An extension to MPI for distributed computing on MPPs", Recent Advances in Parallel Virtual Machine and Message Passing Interface, Lecture Notes in Computer Science, vol. 1332, pp.75-82, November 1997

[7] Y. Suzuki *et al.*, “Research and development of fusion grid infrastructure based on atomic energy grid infrastructure (AEGIS)”, Sixth IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research (4-8 June 2007, Inuyama, Japan), Fusion Engineering and Design, Vol.83, pp.511-515, 2008.

[8] Y. Suzuki *et al.*, “Atomic Energy Grid Infrastructure (AEGIS) and Interoperation with Other Grids”, High Performance Computing on Vector Systems 2008, Springer-Verlag Berlin Heidelberg, pp. 65-77, 2008.

[9] ITER Physics Basis Editors et al., “Chapter 1: Overview and summary”, Nucl. Fusion, vol. 39, pp.2137-2174, 1999.

[10] N. Hayashi, A. Isayama, K. Nagasaki, and T. Ozeki, “Numerical Analysis of Neoclassical Tearing Mode Stabilization by Electron Cyclotron Current Drive”, J. Plasma Fusion Res., vol. 80, pp.605-613, 2004.

[11] N. Hayashi, T. Takizawa, T. Ozeki, N. Aiba, and N. Oyama, “Integrated Simulation of ELM Energy Loss Determined by Pedestal MHD and SOL Transport”, Nucl. Fusion, vol. 47, pp. 682-688, 2007.

[12] K. Hamamatsu, “Numerical Study for Positional Control of ECCD by the Ordinary Wave in a Tokamak Plasma”, J. Plasma Fusion Res., vol. 75, pp.143-150, 1999.

[13] S. Tokuda and T. Watanabe, “A new eigenvalue problem associated with the two-dimensional Newcomb equation without continuous spectra”, Phys. of Plasmas, vol. 6, pp.3012-3026, 1999.

[14] N. Aiba *et al.*, “Numerical Method for the Stability Analysis of Ideal MHD Modes with a Wide Range of Toroidal Mode Numbers in Tokamaks”, Plasma Fusion Res., vol. 2, 010, 2007.

.